Examen final – LOG3000 – Hiver 2014

- Lundi le 28 avril 2014.
- Durée : 13h30 à 16h00 (total 2h30).
- Local : A-532.
- Total des points : 20.
- Pondération de l'examen dans la note finale : 40%.
- Sans documentation.
- ~~Calculatrice permise.~~ Aucune calculatrice.
- Vous pouvez garder le questionnaire.
- Toutes les images peuvent toutes être en noir et blanc.

1. Question sur l'utilité des processus (2 points)

Présentez deux utilités d'avoir un modèle de processus du développement logiciel (1 point).

Justifiez en quoi ces deux utilités sont bénéfiques pour le développement logiciel (1 point).

Pour l'intégration des nouveaux : Le modèle du processus de développement permet aux nouveaux membres de l'équipe de s'intégrer plus facilement, car ils peuvent voir rapidement quel travail ils ont à faire, quelles ressources sont disponibles, et qui utilisera les résultats de son travail.

Pour le diagnostic de problème : Le modèle du processus de développement sert lors des analyses post-mortem afin d'évaluer ce qui a bien été et moins bien été dans le cours d'un projet. Les problèmes diagnostiqués en fin de projet peuvent être associés à des activités problématiques, des artéfacts manquants, des rôles mal définis, etc.

2. Question sur la gestion de risque (2 points)

Remplissez les cases vides du tableau suivant avec ce qui vous semblerait approprié dans le contexte d'un projet intégrateur de 4^{e} année (1 point).

Risque	Probabilité	Impact	Niveau	Stratégie de mitigation
R1 : Un coéquipier arrête de participer à l'avancement du projet.				
R2 : La librairie externe choisie pour être le cœur du projet ne fonctionne pas.				

Justifiez les stratégies de mitigation choisies (1 point).

Réponse :

Risque	Probabilité	Impact	Niveau	Stratégie de mitigation

R1 : Un coéquipier arrête de participer à l'avancement du projet.	Moyenne	Élevé	Élevé	Mitiger le risque en assurant une dissémination de l'information.
R2 : La librairie externe choisie pour être le cœur du projet ne fonctionne pas.	Faible	Élevé	Moyen	Contourner le risque en prévoyant des solutions de rechange.

Mitiger : En s'assurant que personne ne travaille de manière isolée (en silo), on peut éviter les inconvénients d'un membre de l'équipe qui disparait en fin de projet. Un bon moyen d'assurer la dissémination de l'information au sein de l'équipe est de faire de la programmation en paires (*Pair Programming*) avec des changements réguliers de partenaires.

Contourner : On peut prévoir des librairies de rechanges, ou bien une solution différente qui ne nécessite pas de librairies de ce genre. De cette manière, si le risque devient réel, on peut contourner le problème.

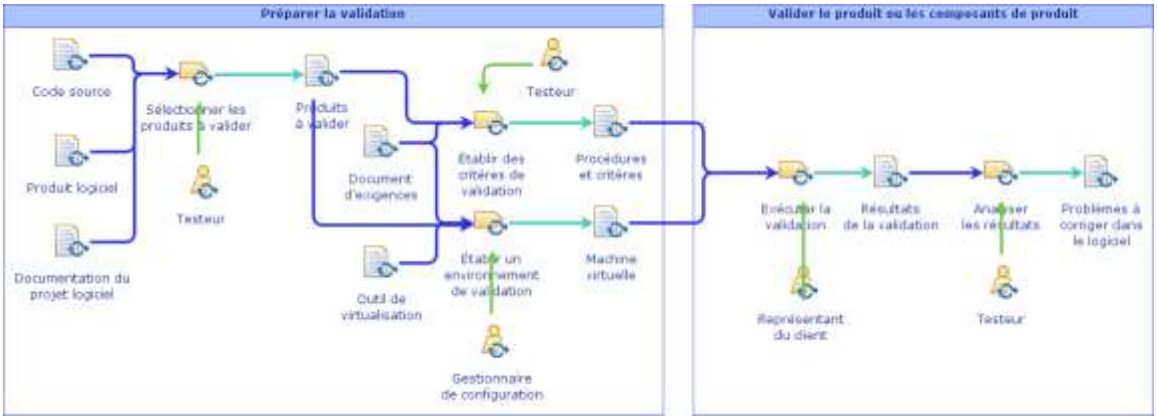
3. Question sur l'application du CMMI (2 points)

Pour le domaine de processus de la validation (VAL), le CMMI demande que les pratiques spécifiques suivantes soient incluses :

- But spécifique 1 : Préparer la validation.
  - Pratique spécifique 1.1 : Sélectionner les produits qui doivent être validés.
  - Pratique spécifique 1.2 : Établir un environnement pour la validation.
  - Pratique spécifique 1.3 : Établir des procédures et critères de validation.
- But spécifique 2 : Valider le produit ou les composants du produit.
  - Pratique spécifique 2.1 : Exécuter la validation.
  - Pratique spécifique 2.2 : Analyser les résultats de la validation.

Modélisez un processus de validation dans le langage SPEM 2.0 tel qu'implémenté dans l'outil ProcessEdit. Ce processus doit répondre aux cinq pratiques spécifiques du CMMI (2 points).

Il y a plusieurs réponses possibles. En voici une :



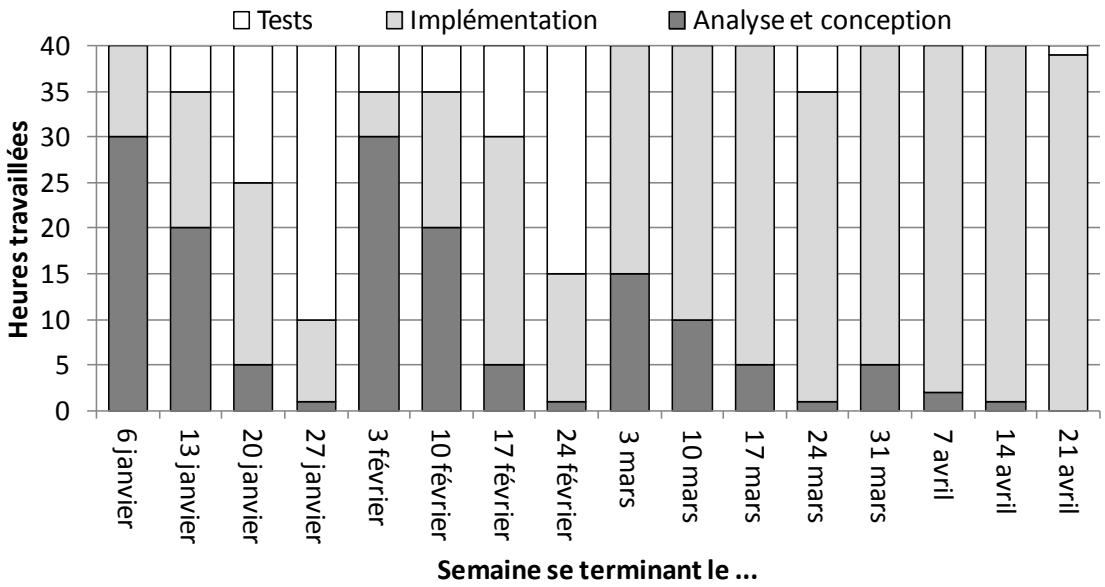
L'important est d'être capable de voir les cinq pratiques du CMMI dans le processus modélisé.

4. *Question d'analyse quantitative (2 points)*

Vous faites une analyse post-mortem d'un projet qui n'a pas donnée de bons résultats. Le logiciel a répondu aux exigences, mais il était tellement mal écrit et rempli de bogues qu'il a dû être abandonné.

Il s'agit d'un petit logiciel écrit par une seule personne durant l'hiver 2014. Cette personne devait travailler à temps plein sur le logiciel (soit 40 heures par semaine). Il n'y avait pas de budget pour du temps supplémentaire, alors le développeur ne pouvait pas y mettre plus de temps.

Le graphe des heures travaillées durant le projet pour les disciplines des tests, d'implémentation et d'analyse et conception sont présentées ci-dessous :



D'après vous, quel est le cycle de vie réel suivi pour produire le logiciel ? Justifiez par des éléments du graphe (1 point).

Il s'agit d'un cycle de vie incrémental. On voit que le cycle dure quatre semaines (ex.: semaine se terminant le 6 janvier jusqu'à la semaine se terminant 27 janvier). Chaque incrément travaille d'abord sur l'analyse et conception, puis sur l'implémentation et finalement sur les tests. On remarque qu'il n'y a pas d'activité d'exigences, ce qui suppose qu'il y a un document d'exigence existant relativement complet : C'est un autre indice d'un cycle de vie incrémental.

Présentez et justifiez un problème potentiel qui aurait pu mener à la mauvaise qualité du code sur la base des données présentées dans le graphe (1 point).

Il semble que les itérations étaient trop courtes. Les deux premières itérations sont relativement complètes, avec une bonne dose d'effort en analyse et conception et en tests. À mesure que le

temps avance par contre, le temps en analyse et conception diminue drastiquement et le temps des tests est virtuellement éliminé. On peut donc supposer que la conception du logiciel est déficiente, ce qui explique son abandon. L'abondance de bogues est compréhensible étant donné que les incréments étaient peu testés.

5. Question sur l'utilité des standards (2 points)

Justifiez deux utilités des standards en décrivant comment les standards permettent d'avoir de meilleurs résultats durant le développement logiciel.

Voici les utilités présentées durant le cours :

- Regrouper les meilleures pratiques : Les standards représentent des ensembles de pratiques reconnues dont l'objectif est d'éviter les erreurs passées. Le respect d'un standard peut permettre d'éviter des crises de gestion en évitant des erreurs communes durant le développement logiciel.
- Encadrer l'assurance-qualité : L'assurance-qualité est une discipline complexe, et les standards détaillent toutes les activités, les artéfacts et les rôles important pour la réalisation d'un produit de qualité.
- Assurer la continuité : Il est plus facile d'intégrer les nouveaux développeurs quand l'organisation utilise des standards reconnus.
- Facilite le partenariat : Un standard permet d'avoir un langage commun pour la communication avec des équipes provenant de d'autres pays, d'autres entreprises, d'autres cultures, etc. Cela permet d'assurer que toutes les informations nécessaires sont échangées, et donc que les équipes font le travail adéquat.

6. Question sur la planification du travail en équipe (3 points)

Vous travaillez sur un petit projet dans une entreprise de taille moyenne. Le projet est un outil qui permet aux vendeurs sur le terrain de documenter leurs ventes sur le serveur central de l'entreprise grâce à leur téléphone intelligent. Le projet implique les personnes suivantes :

- Une équipe de développement de deux personnes (vous-mêmes et un autre développeur) : Vous êtes tous les deux à temps plein sur le projet dans un même bureau à Montréal.
- Une administratrice de système (SysAdmin) qui s'occupe aussi des bases de données (DBA, *Database Administrator*) et qui est responsable du serveur central : Elle est disponible trois jours par semaine sur le projet, mais elle est basée dans un autre bureau à Laval.
- Un vendeur : Il n'est disponible qu'un jour par semaine et peut se retrouver n'importe où au Québec ce jour-là.

Il y a trois parties majeures que le logiciel doit créer, soit :

- L'interface graphique qui doit apparaître sur le cellulaire,

- La logique interne du logiciel qui interagit entre l'interface et la base de données,
- La base de données des ventes faites sur le terrain.

Vous devez donc planifier comment l'équipe va interagir dans le cadre du projet. Définissez une semaine typique de travail (lundi au vendredi) pour les quatre personnes impliquées dans le projet :

- Définissez qui va travailler avec qui sur quelle partie du projet pour chaque jour (1 point).
- Définissez quelles rencontres sont nécessaires : Quand cette rencontre doit avoir lieu, qui doit être présent, comment doit-elle se réaliser (en face-à-face, avec Skype, etc.) et de quoi il sera question (1 point).
- Justifiez votre approche de planification du travail en équipe (1 point).

Il y a plusieurs réponses possibles. En voici une :

- Lundi : En matinée, rencontre des deux développeurs et du vendeur, que l'on contacte à travers un outil comme Skype s'il est hors de Montréal, ou en personne si possible. En après-midi, rencontre en personne des deux développeurs avec l'administratrice de systèmes.
- Mardi : Les deux développeurs travaillent en paires sur l'interface graphique en matinée et sur la logique en après-midi. L'administratrice de systèmes travaille sur la base de données.
- Mercredi : Le premier développeur travaille sur l'interface graphique. Le deuxième développeur travaille sur la logique. L'administratrice de système travaille sur la base de données.
- Jeudi : Les deux développeurs travaillent en paires sur l'interface graphique en matinée et sur la logique en après-midi.
- Vendredi : Le premier développeur travaille sur la logique. Le deuxième développeur travaille sur l'interface graphique. En après-midi, une intégration est faite par les deux développeurs afin d'avoir un produit fonctionnel lorsque c'est possible.

Il y a deux rencontres formelles hebdomadaires de prévues :

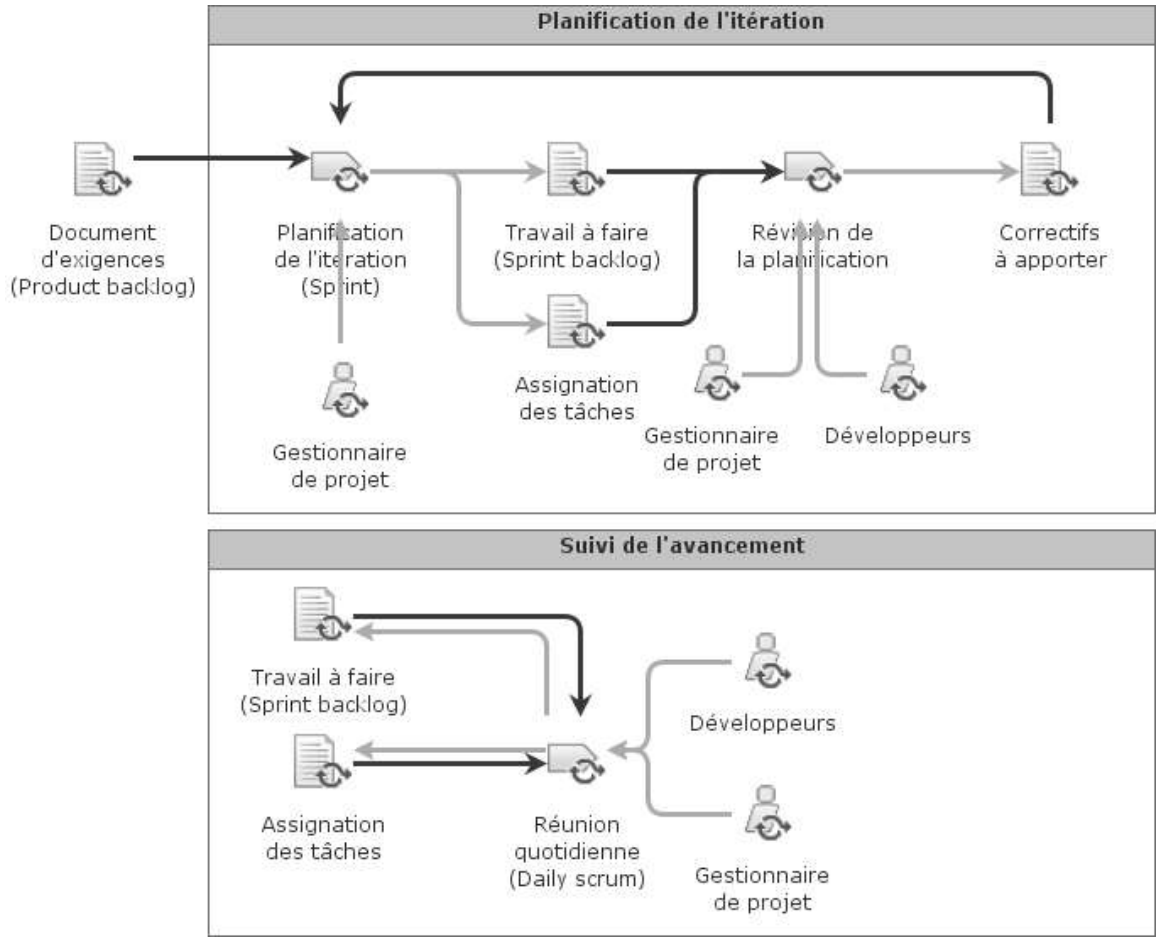
- La rencontre du lundi avant-midi a pour but de présenter au vendeur les travaux effectués et de valider le produit fonctionnel. On obtient du vendeur des rétroactions sur les changements à apporter.
- La rencontre du lundi après-midi a pour but de synchroniser les besoins du logiciel avec les bases de données manipulées. Une liste de changement aux bases est apportée lors de cette réunion afin que ces changements soient faits dans le courant de la semaine.

La planification se concentre sur le travail en paires et la rotation des tâches entre le module graphique et le module de logique. L'objectif est d'assurer la qualité du code des modules en ayant constamment deux paires d'yeux penchés sur ceux-ci. Cela assure aussi qu'il n'y a pas une

appropriation malsaine du code, ce qui permet moins de résistance lorsqu'un changement majeur doit être effectué.

7. Question sur un processus de gestion de projet (2 points)

Vous travaillez pour une entreprise qui gère ses projets avec une approche Agile adaptée du processus Scrum. À chaque début d'itération (appelée "Sprint" dans le processus Scrum), une planification initiale du travail à faire est montée. Cette planification sélectionne les exigences à implémenter pour la prochaine itération. La planification est ensuite révisée jusqu'à l'obtention d'un consensus entre développeurs et gestionnaire de projet. Le processus utilisé pour la gestion de projet est présenté ci-dessous.



Le problème est que les projets de l'entreprise prennent toujours plus de temps que ce qu'il était prévu au départ. Il faut toujours plus d'itérations pour compléter le projet que ce que l'équipe croyait initialement.

Y a-t-il une raison pour laquelle les projets prennent plus de temps que prévu ? Justifiez (1 point).

Le problème majeur est qu'il n'y a pas de planification globale du travail à faire. Il ne semble pas y avoir de nombre prédéterminé d'itérations avec le contenu prévu pour chacune d'entre elle. Il s'ensuit que les estimés du nombre d'itérations sont probablement très arbitraires, et donc souvent erronés.

Proposez une pratique qui permettrait de résoudre ce problème (1 point). Une pratique est définie comme étant forme d'au moins un rôle, une activité, un artéfact en entrée et un artéfact en sortie.

La pratique serait la suivante :

- Activité : Planification globale du projet.
- Artéfact en entrée : Document d'exigences (Product backlog).
- Artéfact en sortie : Fractionnement des exigences en groupes homogènes formant un ensemble précis d'itérations.
- Rôle : Gestionnaire de projet.

8. *Question sur la modélisation d'un processus de maintenance (5 points)*

Vous devez monter le processus de maintenance du projet à code ouvert (*open source*) de Netbeans. Netbeans est un environnement de développement (IDE, *Integrated Development Environment*) créé pour programmer en Java. La principale difficulté provient du fait que les exigences sont difficiles à obtenir car elles proviennent des besoins des utilisateurs de l'IDE.

Le processus de maintenance doit donc :

- Amasser une masse suffisante d'exigences provenant des utilisateurs afin de créer une nouvelle version suffisamment importante. Les exigences peuvent être des bogues à résoudre ou de nouvelles fonctionnalités à implémenter.
- Les exigences amassées doivent être implémentées par des programmeurs volontaires et ensuite soumises aux utilisateurs pour approbation.
- Un cycle de vie de maintenance se termine lorsque toutes les exigences choisies pour former la nouvelle version reçoivent l'approbation des utilisateurs.

Pour ce processus, vous devez modéliser les disciplines suivantes :

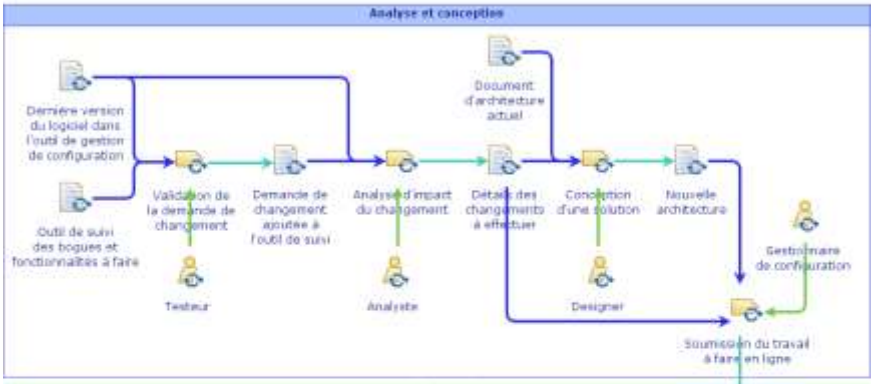
- Analyse et conception (1 point),
- Implémentation (1 point),
- Tests (1 point),
- Gestion de configuration (1 point).

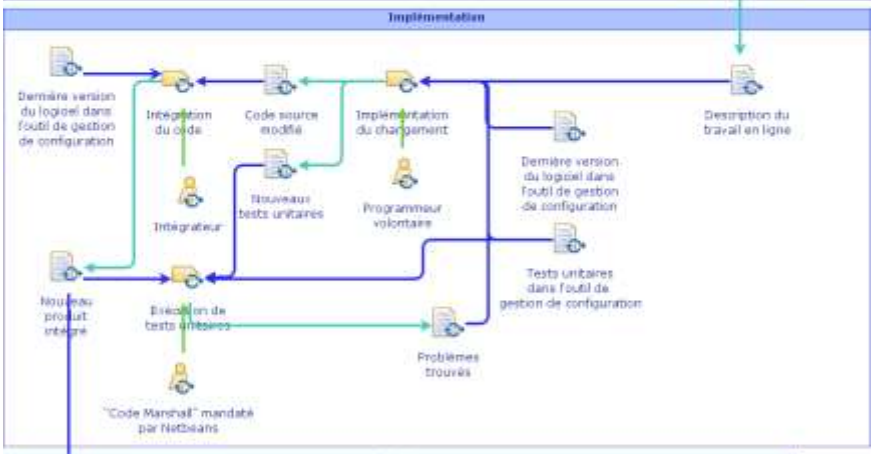
Vous devez aussi choisir un cycle de vie approprié (1 point).

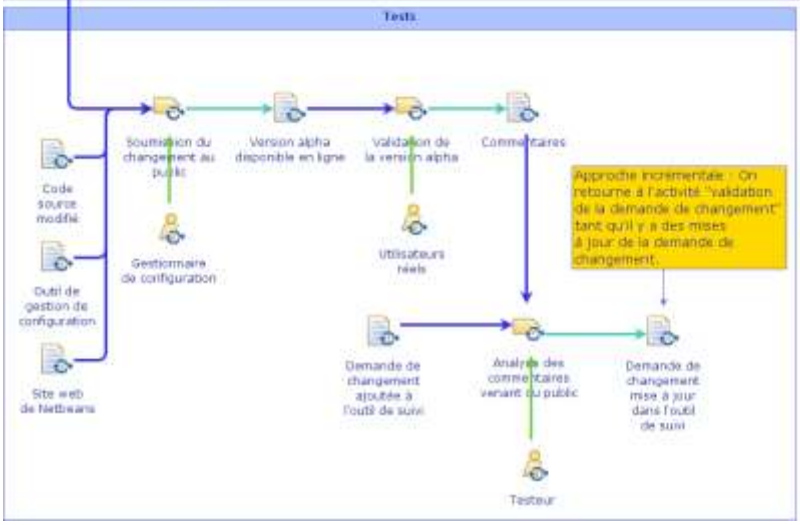
La modélisation doit respecter le langage SPEM 2.0 tel qu'implémenté dans l'outil ProcessEdit. Les erreurs de modélisation seront pénalisées.

Il y a plusieurs réponses possibles. Il faut que le cycle de vie choisi permette une mise à jour des exigences. Idéalement, le cycle de vie doit être transformationnel ou incrémental.


Compilation d'un ensemble de demandes de changements
Cet artefact sert de base pour l'analyse et conception.
Outil de suivi des bogues et fonctionnalités à faire
Dernière version du logiciel dans l'outil de gestion de configuration
Produit logiciel actuel disponible sur le site de Netbeans
Rédaction de commentaires
Utilisateurs réels
Commentaires
Analyse des commentaires venant du public
Analyste
Compilation d'une liste de demandes pour la prochaine version du logiciel
Outil de suivi des bogues et fonctionnalités à faire
Documentation de la nouvelle version dans l'outil de suivi
Gestionnaire de projet


Analyse et conception
Dernière version du logiciel dans l'outil de gestion de configuration
Outil de suivi des bogues et fonctionnalités à faire
Validation de la demande de changement
Testeur
Demande de changement ajoutée à l'outil de suivi
Analyse d'impact du changement
Analyste
Document d'architecture actuel
Détails des changements à effectuer
Conception d'une solution
Designer
Nouvelle architecture
Gestionnaire de configuration
Soumission du travail à faire en ligne


Implémentation
Dernière version du logiciel dans l'outil de gestion de configuration
Intégration du code
Intégrateur
Code source modifié
Nouveaux tests unitaires
Implémentation du changement
Programmeur volontaire
Description du travail en ligne
Dernière version du logiciel dans l'outil de gestion de configuration
Tests unitaires dans l'outil de gestion de configuration
Nouveau produit intégré
Exécution de tests unitaires
"Code Marshall" mandaté par Netbeans
Problèmes trouvés


Tests
Code source modifié
Outil de gestion de configuration
Site web de Netbeans
Soumission du changement au public
Gestionnaire de configuration
Version alpha disponible en ligne
Validation de la version alpha
Utilisateurs réels
Commentaires
Approche incrémentale : On retourne à l'activité "validation de la demande de changement" tant qu'il y a des mises à jour de la demande de changement.
Demande de changement ajoutée à l'outil de suivi
Analyse des commentaires venant du public
Testeur
Demande de changement mise à jour dans l'outil de suivi